Using Evolutionary Multiobjective Techniques for Imbalanced Classification Data

Sandra García, Ricardo Aler, and Inés M. Galván

Computer Science Departament, Carlos III University of Madrid
Avda. Universidad 30, 28911 Leganes, Spain
http://www.evannai.inf.uc3m.es

Abstract. The aim of this paper is to study the use of Evolutionary Multiobjective Techniques to improve the performance of Neural Networks (NN). In particular, we will focus on classification problems where classes are imbalanced. We propose an evolutionary multiobjective approach where the accuracy rate of all the classes is optimized at the same time. Thus, all classes will be treated equally independently of their presence in the training data set. The chromosome of the evolutionary algorithm encodes only the weights of the training patterns missclassified by the NN, instead of all the parameters of the NN as in other approaches. Results show that the multiobjective approach is able to consider all classes at the same time, disregarding to some extent their abundance in the training set or other biases that restrain some of the classes of being learned properly.

Key words: Multiobjective Machine Learning, Imbalanced data, Classification, Neural Networks, NSGA-II.

1 Introduction

Classification is one of the main areas within Machine Learning, whose goal is to learn an input-output mapping function f from a finite set of input-output pairs (x_n, y_n), with discrete y_n, known as the training data. Typically, classification is formulated as an optimization problem: given a family of parameterized functions f_w, the goal is to find the optimal w that minimizes some error or loss function E on the training set (x_n, y_n). For instance, f_w might be the family of Feed-Forward Neural Networks (NN) with a given architecture and a set of connection weights w [3].

Different optimization algorithms can be used, from gradient-based techniques like Backpropagation to optimization methods like Evolution Strategies. However, most of the optimization approaches are single-objective (usually, the training error E is minimized). Yet, classification problems lead very easily to considering several objectives. For instance, in order to avoid overfitting in classification, two objectives can be considered: the error on the training data E_T (to be minimized) and the complexity of the function C (complex functions are

penalized). Commonly, both objectives are combined into a single-objective expression $E = E_T + \lambda * C$, but a new parameter (the regularization parameter λ) must be adjusted. A different approach tries to optimize both objectives at the same time, that is, to consider classification as a Multiobjective Optimization problem[8]. Several works on Evolutionary Multiobjective Machine Learning (EMOML) have shown that generalization can be improved in this way [9]. Other approaches also demonstrate that multiobjective techniques can be used to improve generalization[6, 7].

The aim of this paper is to improve generalization in classification problems for NN where classes are imbalanced. This means that there is much more data for some of the classes than for the rest (see for instance [16]). Usually, learning algorithms that minimize the training error, tend to focus in majority classes at the expense of the rest. In this paper we have found that it is not always the case that it is the minority classes that obtain low accuracies, but it is common that there is some bias against some of the classes. In order to avoid this problem, we propose an evolutionary multiobjective approach where the accuracy rates of all the classes is optimized at the same time. Thus, during learning all classes will be treated equally, independently of their abundance in the training set or other biases. At the end of the learning process, the evolutionary algorithm will produce a Pareto front that represents the tradeoff between the learning rates of the different classes. There remains the issue of selecting this point from the front that it will be discussed in next section.

Our approach is closely related to the optimization of ROC curves. They have been used in Machine Learning as an alternate way of comparing different algorithms, specially if classes are imbalanced or not all classes have the same cost. In a two-class problem, the ROC curve has two dimensions: the true positive rate (TPR) and the false negative rate (FNR). Successful classifiers are the ones whose ROC curves are closer to the top-left point (i.e. highest TPR and lowest FNR). Traditionally, simplified single-objective measures of ROC curves have been used, like the area under the curve (AUC), but EMOML can be applied to optimize directly the ROC curve[1, 5, 6]. The goal of our study is not finding the optimal ROC curve, but to use the accuracy rates as secondary objectives with the aim of indirectly optimize the total accuracy.

So far, it has been described what goals will be optimized (class accuracies) but not what parameters will be explored by the evolutionary algorithm in order to optimize these goals. Typically, EMOML methods codify within the chromosome the parameters of the learning algorithm. The encoding can be either direct or indirect. For instance, a direct encoding for NN would involve encoding the set of real numbers that represent the connection weights w of the network [15]. An indirect encoding would encode a seed in the chromosome from where the topology of the network and/or the weights would be derived by some process (this process is usually called the genotype-phenotype mapping). In the present work, we use an indirect encoding where it is the weights of the training data that are represented on the chromosome. Although the word "weight" is used for both connection weights and training data weights, they are very different concepts.

A weight for a training instance (x_n, y_n) is a real number that tells the learning algorithm about the importance of that particular instance for the learning process. Therefore, in the present work the chromosome encodes the weights for some of the training data. The learning algorithm will use the weighted sample instead of the original dataset. The rationale behind this decision is that it gives the evolutionary algorithm the possibility of focusing in some of the classes by increasing the weights of some of its instances.

The idea of adaptive weights for training data has been successfully used in several areas of Machine Learning, like Boosting[14]. Boosting algorithms iteratively focus on the hard-to-classify instances by increasing the weight of the instances missclassified by the base learning algorithm. In this paper we follow a similar approach: the chromosome will only contain weights for the training instances missclassified by a NN. Otherwise, the chromosomes would have to include a real number for every sample in the training set. Therefore, the number of parameters (weights) would be very large, the optimization process very slow, and the risk of overfitting very high. The chromosome encoding will be explained in more detail in the appropriate section.

Finally, although any classification algorithm could be used, NN have been chosen because it is known that they are sensitive to re-weighting of the training sample. Training will be carried out with Standard Backpropagation which is an algorithm for training multilayer NN. It performs a greedy gradient-based search, trying to minimize the squared mean error evaluated over the training data set. NSGA-II [13, 2] has been chosen as the evolutionary multiobjective optimization algorithm, although others might be used (for instance, in [6] a multiobjective Differential Evolution algorithm is explored).

2 Multiobjective approach for Imbalanced Classification Data

As we mentioned before, we will focus on a specific kind of classification problems known as 'imbalanced class' problems, where the amount of data available for some of the classes is much larger than for the rest. Attempting to minimize the training error on these problems usually results on the minority classes being learned imperfectly. Thus, generalization can be improved by focusing on the minority classes. Of course, improving the accuracy rate of the minority classes can result in worsening the majority classes. Therefore, we have decided to use the multiobjective evolutionary algorithm NSGA-II to look for the optimal Pareto set of NN that represent the best tradeoff between class accuracies. At the end of the evolutionary algorithm, one of the NN will be selected from this set, with the expectation that it improves total generalization and the generalization for minority classes.

In our case, the population of chromosomes of NSGA-II is a set of real numbers to weight the training patterns used by NN. It is clear that if the training set is very large, chromosomes would become very large. Therefore, it has been decided to represent in the chromosome only the weights of the patterns miss-

classified by a NN trained with the original data. Different individuals in the population will be represented by different weightings, that will give rise to different NN after being trained by backpropagation. Each network will focus on the patterns with higher weights. Therefore, our interest is not to search for the connections parameters w of the NN, but to compute how many time each pattern should be replicated.

As NSGA-II must optimize all class accuracies at the same time, our encoding allows NSGA-II to improve some of the classes by replicating some of its patterns. However, this might worsen some of the other classes. Thus, the evolutionary algorithm is in charge of finding the best weighting for all the classes involved. Its output is the best non-dominated set (the front) available at the end of the search. However, we want to produce a single classifier with improved generalization, and not a whole set so the criteria used to select one NN from the front will also be described.

Next, each one of the elements required by NSGA-II will be described more formally.

2.1 The chromosome

The straightforward way to represent a NN is by directly encoding the weights of connections w into the chromosomes. However, in this work, the NN will be represented by the set of data used for training it. More specifically, a chromosome will encode a weighted. Let $|T|$ be the size of the training set, and the training set itself is defined as $T = \{(x_n, y_n), n = 1 : |T|\}$. As mentioned before, the chromosome will only contain the weights of the missclassified samples. Therefore, let NN_T be a NN trained with backpropagation on training sample T. Let $M = \{(x_n, y_n)|y_n \neq NN_T(x_n)\}$ be the set of missclassified patterns, and $T - M$ will be the set of correctly classified patterns.

Then, chromosomes will be defined as an ordered list of real numbers (the weights): $z = z_i, i = 1 : |M|$. Each weight z_i tells how many times pattern i from the missclassified sample M must be replicated. $0 \leq z_i \leq K$, where K is the maximum number of times a pattern can be replicated.

Each chromosome will have an associated NN, constructed by means of backpropagation on a training sample made of the correctly classified patterns $T-M$ and the set of missclassified samples, replicated according to chromosome z. Let's call this replicated sample M_z. Therefore, the NN associated to chromosome z is NN_z constructed from sample $\{T - M, M_z\}$.

2.2 The NSGA-II objectives

Here, the objectives to be optimized by the evolutionary multiobjective optimizer NSGA-II will be defined.

Let C the number of classes in the problem and $|T_c|$ the number of patterns belonging to class c. $T_c = \{(x_i, c), i = 1 : |T_c|\}$ is the set of patterns belonging to class c. As NSGA-II is a minimization algorithm, the class errors will be used

instead of class accuracies. The error for class c is the 0-1 loss for neural network NN_z associated to chromosome z, is defined in Eq.1

$$E_c(z) = \frac{1}{|T_c|} * \sum_{i=1}^{i=|T_c|} \delta(y_i, NN_z(x_i)) \quad (1)$$

It is important to remark that E_c is computed on the original training sample (and not from the replicated sample used to build NN_z). Therefore, the set of goals to be minimized by NSGA-II is $E_c, c = 1 : C$.

2.3 The Fitness Function for NSGA-II

The fitness function evaluates the worth of each of the chromosome z within the population P. In the initial population, the chromosomes have been generated randomly, and they keep being improved during the search process, according to the fitness function. As there are as many objectives as classes, the fitness function has C outputs (Eq. 2).

$$f : z \rightarrow E_1(z), E_2(z), \ldots, E_C(z) \quad (2)$$

The process for computing $f(z)$ will be explained later. But before NSGA-II itself starts, several steps have to be carried out:

- A neural network NN_T is trained on the original training sample T.
- The set of missclassified patterns M is computed. The set of correctly classified patterns is $T - M$.
- The length of chromosomes z in the population P is set to $|M|$. This means that all $z \in P$ are made of $|M|$ real numbers (the weights): $z = z_i, i = 1 : |M|$
- The performance of a NN trained by backpropagation depends on the training data but also on the initial connections weights w_0, which are typically a set of small values generated randomly. This means that during the search process, a chromosome z might outperform another t, not because it is intrinsically better, but because the initial weights. This adds some noise to the fitness function. In order to remove the noise, a set of random initial weigths w_0 is fixed. Therefore, the backpropagation algorithm will start from the same set of initial weights w_0 for all neural networks NN_z created for all chromosomes z during the search process.

Now, the NSGA-II algorithm can start. Initially, all $z \in P$ will be generated randomly by NSGA-II. More specifically, the weights z_i of z will be random numbers, uniformly generated in the range $[0, K]$, where K is a parameter. Every generation, the chromosomes $z \in P$ must be evaluated by the fitness function f, which is computed as follows:

- The replicated sample M_z is computed, according to weights $z_i, i = 1 : |M|$.
- A neural network NN_z is trained with the correctly classified sample and the replicated sample $T - M, M_z$. Backpropagation is used for training, it starts from the set of weights w_0 generated before running NSGA-II.

– Equation 1 is used to compute all objectives $E_c(z)$ (the errors for classes 1 to C). They are numbers between 0 and 1. Those values $E_1(z), E_2(z), \ldots, E_C(z)$ are returned to NSGA-II.

2.4 Selecting a classifier from the non-dominated set

At the end of NSGA-II, a non-dominated set of classifiers $F \in P$ (a front) is obtained. Each point in the front F is a NN, trained with a different sample ($NN_z \in F$). This set of classifiers represents the best tradeoff between objectives $E_1, E_2, \ldots, E_C$ found by NSGA-II. Some of them are good at classifying some of the classes, while others classify correctly other classes. However, our goal is to obtain a single classifier that generalizes well. Standard approaches for model selection could be used[11], to select the classifier that generalizes best from the training set. For instance, the NN from the front that gets the best performance on a separate validation set could be selected. However, this would reduce the size of the training set T, because some data would have to be removed for validation purposes. In the present work, we have selected the classifier that minimizes the total error on the same training set T. This means that the $NN_z \in F$ that obtains the highest value in Eq. 3 will be selected. In case than more than one individual have the same total error, we will choose the one which has the smaller sum of weights.

$$E(z) = \frac{1}{|T|} * \sum_{i=1}^{i=|T|} \delta(y_i, NN_z(x_i)) \tag{3}$$

After that, the generalization of the selected NN_z will be evaluated on a separate test set.

3 Experimental Validation

3.1 Experimental setup

Table 1 shows a summary of the characteristics of the data sets used in this work. The data sets have been selected from the UCI Machine Learning Repository [4] and the Proben Repository [10]. Table 1 displays the number of total training instances and also the number of instances per class in order to identify the imbalanced datasets. Balance-scale, Car, and Thyroid show the largest differences between the majority and the minority classes, whereas Card displays almost no imbalance. Breast-cancer and Ionosphere show some imbalance.

First of all, different architectures of NN have been trained for each domain, in order to select an appropriate number of hidden neurons. In this work, the number of hidden neurons of the NN must be fixed from the start, under the hypothesis that a wide range of architectures will provide similar performance. The objective of this study is to show the advantage of using a multiobjective approach, not to find the optimal architecture for the NN.

Table 1. Classification Domains

Dataset	Number of Attributes	Number of Classes	Number of Patterns Training	Test	Source
Thyroid	21	3	7200	3428	Proben
			166/368/6666		
Car	6	4	1728	-	UCI
			1210/384/69/65		
Balance Scale	4	3	645	-	UCI
			288/49/288		
Breast cancer - W	9	2	699	-	Proben
			241/458		
Ionosphere	34	2	351	-	UCI
			225/126		
Card	51	2	690	-	Proben
			307/383		

All NN are trained with the backpropagation algorithm during 500 iterations and a learning rate of 0.1. They are composed by 3 hidden layers each of them with 15 neurons. The FANN software library has been used [12].

The genes in the NSGA-II chromosome are randomly initialized with real numbers in the interval [0,5]. The population size was set to 30 and NSGA-II was run for 50 generations with a crossover and mutation probabilities of 0.5 and 0.01, respectively.

3.2 Experimental results

In this paper we will use "total classification rate" to refer to the percentage accuracy classification rate of the dataset. "Class classification rate" will be employed to refer to classification rates broken down for each one of the different classes in the problem. Table 2 shows both the total and the class rates obtained by the initial NN for test. 5-fold cross validation has been used for all domains except "Thyroid", because the latter is provided with a test set. Majority and minority classes have been marked with + and −, respectively. It can be observed that in some of the imbalanced domains (Thyroid, Balance-Scale, and Ionosphere), the NN obtains much lower classification rates for the minority classes than for the rest. However, this is not true for Car and Breast Cancer.

The multiobjective algorithm provides a Pareto front with non-dominated individuals. Using the criteria selection described in section 2.4 we choose one individual and check its classification success rate (studying the total one and the rate per each class). These rates are shown in Table 3.

The largest increment in total classification rate (+5.22%) occurs in the Balance Scale data set. This improvement is due to a +57.35% increase in the minority class, without decreasing significantly the rest of the classes. The Car domain also shows a total rate improvement (+1%). In this case, all classes are improved with no particular focus on the minority classes. The accuracy

Table 2. Classification Rate with the Initial Artificial Neural Network

Dataset	Test
	Total Classification Rate (%)
Thyroid	97.81
Car	94.21
Balance Scale	89.14
Breast cancer	96.42
Ionosphere	88.59
Card	82.75
	Class Classification Rates (%)
Thyroid	71.23^{-}/94.31 /98.61^{+}
Car	93.55^{+}/80.41 /88.75^{-}/96.46^{-}
Balance Scale	90.74^{+}/26.98^{-}/99.29^{+}
Breast cancer	96.72^{-}/96.96^{+}
Ionosphere	97.34^{+}/73.74^{-}
Card	79.70^{-}/85.33^{+}

increases range from +2.0% to +6.05%. Let us remember that in Car, the NN obtained higher accuracies for the minority classes, therefore it makes sense that the multiobjective approach will focus on the majority classes. In the Thyroid, and Ionosphere domains, there is some total rate improvement over the initial NN (less than 1%). It does so by focusing mainly on the minority classes. In the Breast Cancer and Card domains there is no significant change over the initial NN results. For the Breast Cancer this is reasonable because it is hard to improve the classification rate for the classes already provided by the NN (96.72% and 96.96% respectively). With regard to Card, the classes were not imbalanced, so the multiobjective approach could not take advantage of focusing in some of the classes in order to improve results.

4 Conclusions

In this paper we have explored a multiobjective evolutionary technique to deal with imbalanced classification problems with Neural Networks (NN). The NN Backpropagation algorithm tends to learn better some of the classes (typically the majority ones) at the expense of the rest of the classes. In order to remove this tendency, we have proposed an evolutionary multiobjective approach that uses NSGA-II, where the accuracy rates of all the classes is optimized at the same time. At the end of the evolutionary process, a Pareto front of NN is obtained. The aim is to improve the accuracies of all the classes but at the same time, increase the total classification rate. The latter is achieved by selecting the NN with maximum total classification rate among the NN in the Pareto front. If there are more than one point with the same total rate, the one with minimum sum of weights was chosen.

In order to generate a diverse front of NN, NSGA-II explored the space of training instance weighs: each input-output pair was associated to a weight.

Table 3. Classification Rate with the Multi-Objective Approach

Dataset	Test
	Total Classification Rate (%)
Thyroid	98.24
Car	95.25
Balance Scale	94.36
Breast cancer	96.70
Ionosphere	89.15
Card	82.46
	Class Classification Rates (%)
Thyroid	72.60^{-}/95.45 /98.99^{+}
Car	97.43^{+}/86.46 /93.57^{-}/98.46^{-}
Balance Scale	92.85^{+}/84.33^{-}/98.13^{+}
Breast cancer	97.38^{-}/95.42^{+}
Ionosphere	97.73^{+}/74.8^{-}
Card	82.23^{-}/82.65^{+}

Thus, NN were trained not on the original sample, but on a weighted sample. The NSGA-II chromosomes contain different weighting sets that give rise to different NN after being trained by Backpropagation on the weighted sample. In order to work with shorter chromosomes, only weights for the training instances missclassified by an initial NN were considered.

The experiments show that in some of the cases, the total classification rate is improved by focusing on the classes that were not learned well by the initial NN. In other cases the total rate was not significantly improved but in general, the accuracy rates of some of the classes improved without decreasing the total classification rate. In summary, the algorithm gives equal opportunity to all classes, independently of their abundance in the training set or independently of other biases, because they are all optimized at the same time.

Acknowledgments. This article has been financed by the Spanish founded research MEC projects OPLINK::UC3M, Ref:TIN2005-08818- C04-02 and MSTAR::UC3M, Ref:TIN2008-06491-C04-03.

5 Bibliography

References

1. Bradley A.P. The use of the area under the roc curve in the evaluation of machine learning algorithms. In *Pattern Recognition*, volume 30, pages 1145–1159, 1997.
2. Kohavi R. Bauer E. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. In *Machine Learning*, volume 36, pages 105–139, 1999.
3. Ripley B.D. *Pattern Recognition and Neural Networks*. Cambridge University Press, 2004.

4. Merz C.J. Blake C.L. Uci repository of machine learning databases. Technical report, University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/mlearn/MLRepository.htm, 1998.
5. Fieldsend J.E. Everson R.M. Multi-class roc analysis from a multi-objective optimisation perspective. In *Pattern Recognition Letters. Special issue: ROC analysis in pattern recognition*, volume 27, pages 918–927, 2006.
6. Martinez F.J. Gutirrez P.A. Cruz M. Fernandez J.C, Hervas C. Memetic pareto differential evolution for designing artificial neural networks in multiclassification problems using cross-entropy versus sensitivity. In *Hybrid Artificial Intelligence Systems*, pages 433–441, 2009.
7. Senhoff B. Graning L., Yaochu J. Generalization improvement in multi-objective learning. In *International Joint Conference on Neural Networks*, 2006.
8. Yaochu J. *Multi-Objective Machine Learning*, pages 151–172. Springer, 2006.
9. Deb K. Knowles J., Corne D. *Multiobjective Problem Solving from Nature. From Concepts to Applications*, pages 155–176. Springer, 2008.
10. Prechelt L. Proben1. a set of neural network benchmarking problems and benchmarking rules. Technical report, Faculty of Computer Science, University of Karlsruhe, Germany, http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/1994/21&search=/ira/1994/21, 1994.
11. Sewell M. Model selection. 2007.
12. Nissen S. Implementation of a fast artificial neural network library (fann). Technical report, Department of Computer Science University of Copenhagen (DIKU), http://leenissen.dk/fann/, 2003.
13. Deb K. Pratap A. Agarwal S. Meyarivan T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. In *IEEE Transactions on Evolutionary Computation*, volume 6, pages 182–197, 2002.
14. Wang H. Wang X. Classification by evolutionary ensembles. In *Pattern Recognition*, volume 39, pages 595–607, 2006.
15. Senhoff B. Yaochu J. Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transanctions on Systems, Man, and Cybernetics part C: Applications and Reviews*, 38:397–415, 2008.
16. Lee A. Feldkamp Yi L. Murphey, Hong Guo. Neural learning from unbalanced data. In *Applied Intelligence*, pages 117–128, 2004.